Estructuras de Datos 1 - ST0245
Examen Parcial 1

Nombre:.....

Departamento de Informática y Sistemas
Universidad EAFIT

Marzo 14, 2017

Criterios de calificación

NOTAS IMPORTANTES:

- Responda en la hoja de PREGUNTAS
- Marque la hoja de PREGUNTAS
- Selección múltiple con única respuesta
  - Respuesta correcta: 100 %
  - Respuesta incorrecta: 0 %
- Completar código
  - Respuesta correcta 100 %
  - Respuesta incorrecta o vacía 0 %

1. Notación O grande 10 %

Supongamos que $P(n, m)$ es una función cuya complejidad asintótica es $O(n \times m)$ y H(n,m) es otra función cuya complejidad asintótica es $O(m * A(n, m))$, donde $A(n, m)$ es otra función. ¿Cuál de las siguientes funciones, definitivamente, **NO** podría ser la complejidad asintótica de $A(n, m)$ si tenemos en cuenta que $P(n, m) > H(n, m)$ para todo n y para todo m?

A $O(log\ n)$
B $O(\sqrt{n})$
C $O(n + m)$
D $O(1)$

2. Complejidad 10 %

Dayla sabe que la complejidad asintótica de la función `P(n)` es $O(\sqrt{n})$. Ayúdale a Dayla a sacar la complejidad asintótica para la función `mistery(n,m)`.

```
void mistery(int n, int m)
  for(int i = 0; i < m; ++i)
    boolean can = P(n);
    if(can)
      for(int i = 1; i * i <= n; i++)
        //Hacer algo en O(1)
    else
      for(int i = 1; i <= n; i++)
        //Hacer algo en O(1)
```

La complejidad de `mistery(n,m)` es:

A $O(m + n)$
B $O(m \times n \times \sqrt{n})$
C $O(m + n + \sqrt{n})$
D $O(m \times n)$

3. Listas 10 %

¿Cuál operación tiene una mayor complejidad asintótica, para el peor de los casos, en un lista simplemente enlazada?

A Buscar un dato cualquiera en la lista
B Insertar un elemento cualquiera en la lista
C Las dos tienen la misma complejidad asintótica

4. Complejidad 10 %

Sabemos que `P(n)` ejecuta $n^3 + n$ pasos y que `D(n)` ejecuta $n + 7$ pasos. ¿Cuál es la complejidad asintótica, en el peor de los casos, de la función `B(n)`?

```
public int B(int n)
  int getP = P(n);
  int ni = 0;
  for(int i = 0; i < n; ++i)
    if(D(i) > 100){
      ni++;
  int nj = getP + D(n) * ni;
  return nj;
```

A $O(n^4)$
B $O(n^3)$
C $O(n^2)$
D $O(2^n)$

5. Colas 10 %

¿Cuál es la complejidad asintótica, para el peor de los casos, de la función `procesarCola(q, n)`?

```
public void procesarCola(Queue q, int n)
  for (int i = 0; i < n; i++)
     for(int j = 0; j < n; j++)
        q.add(j);
        //Hacer algo en O(1)
```

A $O(n)$
B $O(|q|^2)$, $|q|$ es el número de elementos de `q`
C $O(n^2)$
D $O(2^n)$

En Java, el método `add` agrega un elemento al comienzo de una cola.

6. Recursion 10 %

Pepito escribió un algoritmo que, dado un arreglo de enteros, decide si es posible escoger un subconjunto de esos enteros, de tal forma que la suma de los elementos de ese subconjunto sea igual a `target`. El parámetro `start` funciona como un contador y representa un índice en el arreglo de números `nums`.

```
01 public boolean SumaGrupo(int start,
                      int[] nums, int target) {
02  if (start >= nums.length) return target == 0;
03  return SumaGrupo(start + 1, nums,
                           target - nums[start])
04          || SumaGrupo(____,___,___);
05 }
```

¿Qué parámetros colocaría en el llamado recursivo de la línea 4 para que el programa funcione?

`___________`, `___________`, `____________________`

7. Colas 40 %

En el juego de *hot potato* (conocido en Colombia como *Tingo, Tingo, Tango*), los niños hacen un círculo y pasan al vecino de la derecha, un elemento, tan rápido como puedan. En un cierto punto del juego, se detiene el paso del elemento. El niño que queda con el elemento, sale del círculo. El juego continúa hasta que sólo quede un niño.

Este problema se puede simular usando una cola. El algoritmo tiene dos entradas: Una cola `q` con los nombres de los niños y una constante entera `num`. El algoritmo retorna el nombre de la última persona que queda en el juego, después de pasar la pelota, en cada ronda, `num` veces. Como un ejemplo, para el círculo de niños *[Bill, David, Susan, Jane, Kent, Brad]*, donde *Bill* es el primer niño, *Brad* el último, y `num` es igual a 7, la respuesta es *Susan*.

En Java, el método `add` agrega un elemento al comienzo de una cola, y el método `remove` retira un elemento del final de una cola y retorna el elemento.

```
01 String hotPotato(Queue<String> q, int num)
02   while (____________)
03      for (int i = 1; i ____ num; i++)
04         q.add(_________);
05      q.remove();
06   return ___________;
```

A continuación, complete los espacios restantes del código anterior:

A (10 %) Complete el espacio de la línea 02
`______________________`
B (10 %) Complete el espacio de la línea 03
`______________________`
C (10 %) Complete el espacio de la línea 04
`______________________`
D (10 %) Complete el espacio de la línea 06
`______________________`